Stock Price Movement Analysis

By-Ridhima Goyal

202401100300198

INTRODUCTION

The task of stock price prediction has long been a topic of interest among investors and researchers. With the advent of machine learning and data analysis tools, predicting stock movements has become more accessible. However, stock market predictions are inherently challenging due to their volatile and dynamic nature.

This document introduces a Rule-Based Stock Price Prediction Algorithm that uses historical stock price data to generate buy or sell signals based on technical indicators. The approach focuses on the use of Simple Moving Averages (SMA), a widely known and effective tool in technical analysis, to predict the movement of stock prices. The algorithm is designed to analyze historical stock data and generate predictions based on the following rules:

- **Buy Signal: When the short-term moving average (50-day SMA) is above the long-term moving average (200-day SMA) and the price change for the day is positive (the stock price increased).**
- **Sell Signal: When the short-term moving average (50-day SMA) is below the long-term moving average (200-day SMA) and the price change for the day is negative (the stock price decreased).**
- **No Action: When the stock price movement does not meet the conditions for a buy or sell signal.**

The program utilizes Python, specifically the pandas library, to read historical stock data from a CSV file, calculate relevant technical indicators, and apply the rule-based prediction system. The results are then visualized using matplotlib to provide an easy-to-understand representation of stock price movements and the algorithm's predictions.

This method offers a straightforward and interpretable approach to stock price prediction, making it an ideal starting point for those looking to explore stock market analysis using rule-based techniques. However, it is important to note that this method relies purely on historical data and does not consider external factors, news, or market sentiment, which may have a significant impact on stock prices.

Methodology

The methodology for the Rule-Based Stock Price Prediction Algorithm follows a systematic process that involves the collection, preprocessing, analysis, and evaluation of historical stock price data. The key components of the methodology are outlined below:

---

1. Data Collection and Import

The first step involves acquiring historical stock price data. In this methodology, stock price data is imported through a CSV file, which can be obtained from various financial sources such as Yahoo Finance, Alpha Vantage, or Quandl. The data typically contains columns such as Date, Open, High, Low, Close, and Volume.

- **Data Import: The stock price data is uploaded and read into a pandas DataFrame, allowing for easy manipulation and analysis.**

python

Copy

```
df = pd.read_csv(file_name)
```

- **Data Inspection: The initial data is inspected to check for missing values, correct data types, and consistency.**

python

Copy

```
df.head()
```

---

2. Feature Engineering

Feature engineering is the process of transforming raw data into meaningful indicators that help in predicting stock movements. In this case, technical indicators such as Simple Moving Averages (SMA) and Price Change are calculated:

- **Simple Moving Averages (SMA):**
    - **The 50-day SMA and 200-day SMA are calculated to represent short-term and long-term trends in stock price movements. These are commonly used technical indicators in the stock market.**

python

Copy

```
df['SMA_50'] = df['Close'].rolling(window=50).mean()  # Short-term trend
df['SMA_200'] = df['Close'].rolling(window=200).mean()  # Long-term trend
```

- **Price Change:**
    - **The daily percentage change in stock price is calculated using the closing price for each day. This helps track how much the stock price fluctuates on a daily basis.**

python

Copy

```
df['Price_Change'] = df['Close'].pct_change()  # Percentage change in price
```

These features are then added to the DataFrame to aid in predicting stock price movements.

---

3. Rule-Based Prediction Algorithm

The core of the methodology is the rule-based prediction system. The algorithm applies a set of simple rules based on the calculated features to generate buy or sell signals.

- **Prediction Rules:**
  - **Buy Signal: If the short-term moving average (50-day SMA) is above the long-term moving average (200-day SMA) and the stock price change for the day is positive (price increases), then predict a buy signal (1).**
  - **Sell Signal: If the short-term moving average (50-day SMA) is below the long-term moving average (200-day SMA) and the stock price change for the day is negative (price decreases), then predict a sell signal (-1).**
  - **No Action: If neither condition is met, the algorithm predicts no action (0).**

python

Copy

```
def predict_price_movement(row):
    if row['SMA_50'] > row['SMA_200'] and row['Price_Change'] > 0:
        return 1  # Buy Signal
    elif row['SMA_50'] < row['SMA_200'] and row['Price_Change'] < 0:
        return -1  # Sell Signal
    else:
        return 0  # No Action
```

The prediction is applied to each row in the dataset using the apply() method, which ensures that every day's stock data is evaluated according to the rules.

---

4. Prediction Evaluation

Once the predictions are generated, the performance of the algorithm is evaluated by comparing the predicted signals with the actual market movement. This evaluation helps measure the effectiveness of the rule-based system.

- **Actual Market Movement:**
    - **The actual movement is determined by comparing the closing price of the current day with the next day's closing price:**
        - **Price Up: If the next day's closing price is higher than the current day's closing price, the actual movement is considered an upward movement (1).**
        - **Price Down: If the next day's closing price is lower than the current day's closing price, the actual movement is considered a downward movement (-1).**

python

Copy

```
df['Next_Day_Close'] = df['Close'].shift(-1)
df['Actual_Movement'] = np.where(df['Next_Day_Close'] > df['Close'], 1, -1)
```

- **Accuracy Calculation:**

- The accuracy of the model is computed by comparing the predicted movement with the actual movement. The prediction is considered correct if the predicted buy or sell signal matches the actual market movement.

python

Copy

```
accuracy = (df['Prediction'] == df['Actual_Movement']).sum() / len(df) * 100
```

The higher the accuracy, the better the model is at predicting stock price movements.

---

5. Visualization

To visually assess the predictions, the algorithm plots the stock price, moving averages, and the generated buy and sell signals on a graph. This helps in understanding how well the model's predictions align with actual market movements.

- **Plotting:**
  - **The stock price and the moving averages are plotted over time.**
  - **Buy signals (green markers) and sell signals (red markers) are overlaid on the stock price graph.**

python

Copy

```
plt.scatter(buy_signals['Date'], buy_signals['Close'], marker='^', color='g', label='Buy Signal', alpha=1)
```

plt.scatter(sell_signals['Date'], sell_signals['Close'], marker='v', color='r', label='Sell Signal', alpha=1)

This step provides a visual representation of the algorithm's performance, allowing users to quickly identify when the algorithm generated buy or sell signals relative to stock price movements.

---

6. Conclusion

This methodology provides a simple yet effective approach for predicting stock price movements using a rule-based algorithm. By leveraging Simple Moving Averages (SMA) and price changes, the model generates buy and sell signals. The methodology also includes steps for evaluating the model's accuracy, as well as visualizing the predictions for further analysis.

While this rule-based system provides a basic framework, it can be expanded by incorporating additional technical indicators, more complex rules, and even machine learning techniques for improved accuracy and decision-making in stock trading.

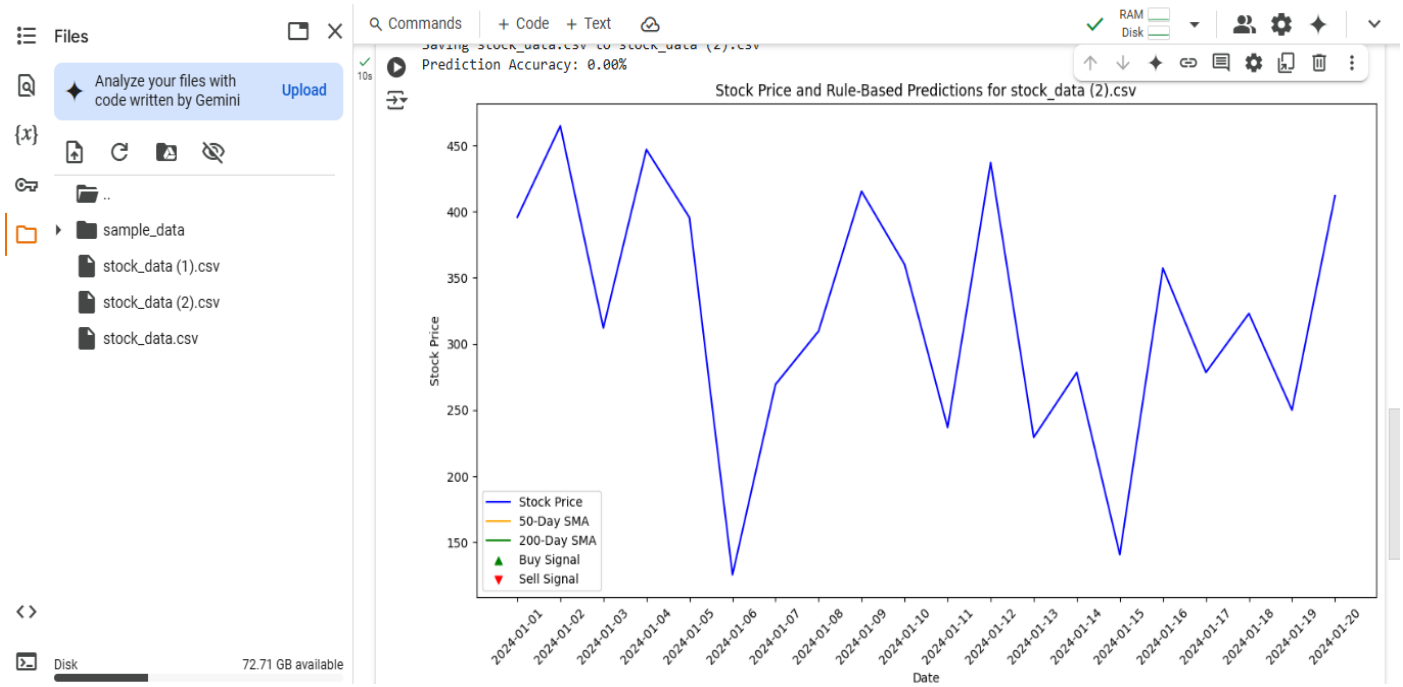
CODE TYPED

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Step 1: Upload file in Google Colab
from google.colab import files

# Upload a CSV file with stock price data
uploaded = files.upload()

# Load the dataset into a pandas DataFrame
# Assuming the file has columns like 'Date', 'Open', 'High', 'Low', 'Close', 'Volume'
file_name = list(uploaded.keys())[0]  # Get the uploaded file name
df = pd.read_csv('/content/stock_data.csv')

# Check the first few rows to understand the structure of the dataset
df.head()

# Step 2: Feature Engineering (Calculating Technical Indicators)
```

```
df['SMA_50'] = df['Close'].rolling(window=50).mean()  # 50-day Simple Moving Average
df['SMA_200'] = df['Close'].rolling(window=200).mean()  # 200-day Simple Moving Average
df['Price_Change'] = df['Close'].pct_change()  # Percentage change in stock price

# Step 3: Define Rule-based Algorithm to Predict Price Movement
def predict_price_movement(row):
    if row['SMA_50'] > row['SMA_200'] and row['Price_Change'] > 0:
        return 1  # Predict price increase (Buy signal)
    elif row['SMA_50'] < row['SMA_200'] and row['Price_Change'] < 0:
        return -1  # Predict price decrease (Sell signal)
    else:
        return 0  # No movement or uncertain

# Apply the rule-based prediction
df['Prediction'] = df.apply(predict_price_movement, axis=1)

# Step 4: Evaluate the Predictions
# Shift the 'Prediction' column by 1 to compare the actual movement of the next day
df['Next_Day_Close'] = df['Close'].shift(-1)
```

```
df['Actual_Movement'] = np.where(df['Next_Day_Close'] > df['Close'], 1, -1)

# Calculate accuracy: Comparing prediction with actual movement
accuracy = (df['Prediction'] == df['Actual_Movement']).sum() / len(df) * 100
print(f"Prediction Accuracy: {accuracy:.2f}%")

# Step 5: Visualize the Results
plt.figure(figsize=(14, 7))

# Plotting the stock price and moving averages
plt.plot(df['Date'], df['Close'], label='Stock Price', color='blue')
plt.plot(df['Date'], df['SMA_50'], label='50-Day SMA', color='orange')
plt.plot(df['Date'], df['SMA_200'], label='200-Day SMA', color='green')

# Highlight Buy and Sell signals
buy_signals = df[df['Prediction'] == 1]
sell_signals = df[df['Prediction'] == -1]

plt.scatter(buy_signals['Date'], buy_signals['Close'], marker='^', color='g', label='Buy Signal', alpha=1)
```

```
plt.scatter(sell_signals['Date'], sell_signals['Close'], marker='v',
color='r', label='Sell Signal', alpha=1)


plt.xlabel('Date')

plt.ylabel('Stock Price')

plt.title(f"Stock Price and Rule-Based Predictions for {file_name}")

plt.legend()

plt.xticks(rotation=45)

plt.show()
```


Files
Analyze your files with code written by Gemini
Upload
sample_data
stock_data (1).csv
stock_data (2).csv
stock_data.csv
Disk
72.71 GB available
Commands
+ Code
+ Text
RAM
Disk
Prediction Accuracy: 0.00%
Stock Price and Rule-Based Predictions for stock_data (2).csv
Stock Price
Stock Price
50-Day SMA
200-Day SMA
Buy Signal
Sell Signal
Date